UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/602,515	06/23/2000	Leonard J. Testa	4675-008	9084

4678 7590 03/16/2005

MACCORD MASON PLLC
300 N. GREENE STREET, SUITE 1600
P. O. BOX 2974
GREENSBORO, NC 27402

EXAMINER	
VAN DOREN, BETH	

ART UNIT	PAPER NUMBER
3623	

DATE MAILED: 03/16/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/602,515	TESTA, LEONARD J.	
	Examiner	Art Unit	
	Beth Van Doren	3623	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 27 January 2005.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-54 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-54 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Continued Examination Under 37 CFR 1.114

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application on 01/27/05. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 07/15/04 has been entered.

2. Claims 1, 27, 28, 53, and 54 have been amended. Claims 1-78 are now pending in this application.

Claim Rejections - 35 USC § 112

3. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

Claims 1, 27, and 28 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

4. Claim 1 recites "a scheduling system for scheduling a plurality of time-dependent tasks" in the preamble. Step (d) recites "to generate a schedule for selecting said plurality of time dependent tasks". Therefore, the recitation in the preamble indicates that the system is for scheduling the plurality of tasks, whereas step (d) recites that the schedule is for selecting said tasks, and therefore not all the tasks are being scheduled. Based on this discrepancy, it is unclear as to what is specifically occurring in the claim. Claim 28 recites equivalent recitations and therefor has the same deficiencies. Clarification is required in both instances.

5. Claim 27 recites "a dynamic programming [...] adapted to provide at least one solution for a scheduling problem". However, steps a and b in the body of claim 27 do not accomplish this providing of a solution. Step a states that a hashing function is capable of detecting duplicate solutions generated and step b recites a binary tree that provides searching, insertion, and deletion on the lowest-cost partial paths. However, it is unclear as to what the scheduling problem is and how the solution is generated. Further, there is no step that recites any providing of the solution. Therefore, it appears as if a step has been omitted. Correction is required.

Claim Rejections - 35 USC § 101

6. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 1-54 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

The basis of this rejection is set forth in a two-prong test of (1) whether the invention is within the technological arts and (2) whether the invention produces a useful, concrete, and tangible result. For a claimed invention to be statutory, the claimed invention must be within the technological arts. Mere ideas in the abstract that do not apply, involve, use, or advance the technological arts fail to promote the "progress of science and the useful arts" (i.e. the physical sciences as opposed to social sciences, for example) and therefore are found to be non-statutory subject matter. For a process claim to be considered statutory, the recited process must somehow apply, involve, use, or advance the technological arts.

In the present case, claim 1 recites a system comprised of modules. Examiner points out that a module taken in its broadest reasonable interpretation is merely a component of a system and since the claim never specifically embodies the system in a computer readable medium, claim 1 does not necessarily involve and apply the technological arts. Further, even if it is assumed that the system is a computer system, a module is merely a portion of a program and therefore a system comprising modules is software per se. Software is respectfully not considered statutory subject matter. Claims 2-26 depend on claim 1 and therefore contain the same deficiency. Claim 27 claims a module with a hashing function and a height-balanced binary tree. For the reasons stated above a module is not considered statutory subject matter without embodiment in computer executable medium. Claim 28 is also a system comprised of modules, with claims 29-52 dependent thereon, and therefore is not considered statutory for the same reasons as set forth above with regards to claim 1. Claims 53 and 54 recite a partitioner module and a scheduling module and generating a schedule. However, in its broadest reasonable interpretation, a module is merely a component and since the claims never specifically embody the methods in computer readable mediums, claims 53 and 54 do not necessarily apply and involve the technological arts.

Claims 1-54 do produce a useful, concrete, and tangible result. However, since the claimed invention is not within the technological arts, as explained above, claims 1-54 are deemed to be directed towards non-statutory subject matter.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-26 and 28-54 are rejected under 35 U.S.C. 103(a) as being unpatentable over Syswerda (U.S. 5,319,781) in view of Oba et al. (U.S. 5,241,465).

8. As per claims 1 and 28, Syswerda teaches a scheduling system for scheduling a plurality of time-dependent tasks, said scheduling system comprising:

(a) an enumerative brute force module (See figure 5, column 2, lines 39-40, and column 5, lines 62-67, wherein tasks are listed using no intelligence or reasoning);

(b) a deterministic module (See figures 5 and 7, column 2, lines 40-45, column 3, lines 20-40, column 4, lines 20-36, column 6, lines 1-10, and column 8, lines 1-7, that considers the constraints to determine an ordering of the tasks);

(c) a genetic module (See figure 5, column 3, lines 50-67, column 4, lines 50-67, column 6, lines 14-35, wherein a genetic algorithm is implemented);

(d) a partitioner module for selecting one of said brute force module, said dynamic programming module, or said genetic module to generate a schedule for selecting said plurality of time-dependent tasks (See figures 5 and 7, column 2, lines 39-45, column 3, lines 20-40 and 50-67, column 4, lines 20-36, column 5, lines 62-67, column 6, lines 1-10 and 14-35, column 8, lines 1-7, and the appendix, which discloses a computer implementation which gives a portion ability to divide the scheduling responsibilities by selecting one of the modules. This portion has control over the flow of the schedule generations); and

(e) a constraint file, said constraints file providing an input to said partitioner module (See column 3, lines 20-40, and appendix, which discloses storing constraints in the computer implemented scheduling system).

(f) the functionality of grouping schedules by score and deleting those schedules grouped at specific score thresholds (See column 4, lines 20-32 and 60-67, column 5, lines 1-5, and column 6, lines 14-40, wherein schedules are evaluated, grouped by score, and deleted at a certain threshold score. Since duplicate schedules would attain the same score when evaluated by the system, duplicates would be grouped together and duplicates would be deleted).

However, Syswerda does not expressly disclose a dynamic programming module that generates lowest-cost partial paths for said plurality of time-dependent tasks, said dynamic programming module including a hashing function having a low probability of collisions and a height-balanced binary tree for providing search insertion and deletion operations on said lowest-cost partial paths.

Oba et al. discloses a dynamic programming module that generates lowest-cost partial paths for said plurality of time-dependent tasks with a height-balanced binary tree for providing search insertion and deletion operations (See at least figure 10, column 1, lines 43-45 and 62-67, column 2, lines 1-5, column 4, lines 1-13, column 5, lines 10-34 and 48-65, and column 8, lines 32-65, wherein a height-balanced binary tree provides for search insertion and deletion operations and lowest cost consideration (cost being time, resource, money)).

However, Oba et al. does not expressly disclose a hashing function having a low probability of collisions.

Syswerda teaches a scheduling system that generates schedules by functionally selecting between scheduling modules available in the system, these modules including genetic, enumerative brute force, and deterministic scheduling techniques. Syswerda et al. also discloses when discussing the deterministically programmed schedule builder that this module can include a more complex scheduling techniques, as stated in column 6, lines 5-11, and column 7, lines 29-37. Oba also teaches a scheduling system that generates, evaluates, and eliminates schedules to generate optimal schedules based on the constraints, including a dynamic programming module that generates lowest-cost partial paths with a height-balanced binary tree. Dynamic programming considers changeable competing or conflicting forces, such as resource and time constraints, to produce multiple possible outcomes and therefore would be considered a specific type of deterministic scheduling by one of ordinary skill in the art. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to use the framework of the system of Syswerda (a scheduler that has multiple scheduling functions available to it) and include the dynamic programming function taught by Oba et al. as the deterministic module of Syswerda in order to increase the ability of the system to build better schedules while still acting deterministically, as stated in column 7, lines 29-40.

Furthermore, Syswerda teaches grouping schedules by score and deleting those schedules grouped at specific score thresholds as well as deterministically building schedules by applying soft and hard constraints to lists of tasks. Hashing functions are well known in the computer programming art and are used to identify and map values to a location for faster data retrieval. It is also known in the art of programming that a hash function should be fast and it should cause as few collisions as possible so that it does not produce the same value from two different inputs. It

would have been obvious to one of ordinary skill in the art at the time of the invention to use a hashing function to group the schedules in order to more efficiently map schedules of duplicate task lists and/or duplicate scores to the same location for easy deletion.

9. As per claim 2, Syswerda discloses a scheduling system further comprising a constraints file, said constraints file providing an input to said partitioner module (See column 3, lines 20-40, which discloses storing constraints in the computer implemented scheduling system).

10. As per claims 3 and 29, Syswerda teaches wherein said constraints file includes at least one predetermined time constraint (See column 3, lines 20-40, column 4, lines 20-40, column 6, lines 1-10, which discloses a predetermined time constraint).

11. As per claims 4 and 30, Syswerda teaches wherein said predetermined time constraint further includes at least one constraint selected from the group consisting of:

(a) a precedence constraint, wherein said precedence constraint limits scheduling of said tasks according to a predetermined order of occurrence in time (See column 3, lines 20-40, column 4, lines 20-40, column 6, lines 1-10, which includes at least one of a precedence constraint, a time window constraint, a priority constraint, or a conditional constraint);

(b) a time window constraint, wherein said time window constraint limits scheduling of said tasks within a time window of a predetermined length (See column 3, lines 20-40, column 4, lines 20-40, column 6, lines 1-10, which includes at least one of a precedence constraint, a time window constraint, a priority constraint, or a conditional constraint);

(c) a priority constraint, wherein said priority constraint limits scheduling of said tasks to a sequence according to a predetermined priority (See column 3, lines 20-40, column 4,

lines 20-40, column 6, lines 1-10, which includes at least one of a precedence constraint, a time window constraint, a priority constraint, or a conditional constraint); and

(d) a conditional constraint, wherein said conditional constraint limits scheduling of said tasks based on an occurrence of at least one event (See column 3, lines 20-40, column 4, lines 20-40, column 6, lines 1-10, which includes at least one of a precedence constraint, a time window constraint, a priority constraint, or a conditional constraint).

12. As per claims 5 and 31, Syswerda discloses wherein the brute force module includes a task list and a schedule permutation generator and Syswerda discloses a schedule evaluator (See figure 5, column 2, lines 39-40, and column 5, lines 62-67, wherein tasks are listed in random order using no intelligence or reasoning. See at least figure 5, column 3, lines 40-49, column 4, lines 36-50, and column 6, lines 1-13, wherein evaluating the schedules produced is disclosed). However, Syswerda does not expressly disclose a schedule evaluator in the brute force module.

Syswerda discloses a brute force module that generates random schedules of tasks from the task list as well as evaluating these randomly generated schedules through the application of constraints and evaluations. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to include the evaluation functionality in the brute force module in order to increase the efficiency of the system by evaluating the generated schedules at each stage of the process, thereby removing excess schedules and allowing fewer and better schedules to proceed to the next process.

13. As per claims 6 and 32, Syswerda discloses wherein said schedule permutation generator includes a first algorithm, said first algorithm enumerating each possible permutation of the time

dependent tasks in a schedule (See column 5, lines 62-67, wherein the permutation generator creates schedules with complete lists including all the tasks).

14. As per claims 7 and 33, Syswerda teaches wherein said schedule evaluator is an efficiency evaluator (See figure 5, column 3, lines 20-30 and 40-49, column 4, lines 36-50, and column 6, lines 1-13, wherein the schedule evaluator is discussed that evaluates the ability of the system to produce a desired effect).

15. As per claims 8 and 34, Syswerda discloses wherein said efficiency evaluator includes at least one parameter selected from the group consisting of a time parameter, a cost parameter, and a user-defined parameter (See figure 5, column 3, lines 20-30 and 40-49, column 4, lines 36-50, and column 6, lines 1-13, wherein the evaluator evaluates on a parameter including at least one of time, cost, or user-defined).

16. As per claims 9 and 35, Syswerda teaches wherein said efficiency evaluator includes a second algorithm, said second algorithm determining an efficiency of each task based on a position of said task in a schedule (See column 5, lines 62-67, and column 6, lines 1-13, wherein each task of the randomly generated schedule is ordered and placed in its first legal position based on the constraints and then the schedule is evaluated based on the ordering).

17. As per claims 10 and 36, Syswerda discloses wherein said deterministic module includes a task list and a schedule permutation generator and Syswerda also discloses a schedule evaluator (See figures 5 and 7, column 2, lines 40-45, column 3, lines 20-40, column 4, lines 20-36, column 5, lines 62-67, and column 6, lines 1-13, and column 8, lines 1-7, wherein a deterministic module is disclosed that includes a task list and a schedule permutation generator. See at least figure 5, column 3, lines 40-49, column 4, lines 36-50, and column 6, lines 1-13, wherein

evaluating the schedules produced is disclosed). However, Syswerda does not expressly disclose a dynamic programming module or a schedule evaluator in the module.

Syswerda discloses a deterministically programmed schedule builder that obtains a list of tasks and orders them using hard and soft constraints. Syswerda et al. also discloses when discussing the deterministically programmed schedule builder that this module can include a more complex scheduling techniques, as stated in column 6, lines 5-11, and column 7, lines 29-37. Dynamic programming considers changeable competing or conflicting forces, such as resource and time constraints, to produce multiple possible outcomes and therefore would be considered a specific type of deterministic scheduling by one of ordinary skill in the art. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to use dynamic programming techniques in the deterministic module of Syswerda in order to increase the ability of the system to build better schedules while still acting deterministically, as stated in column 7, lines 29-40.

Furthermore, Syswerda discloses a module that generates schedules of tasks from the task lists as well as evaluating these generated schedules through the application of constraints and evaluations. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to include the evaluation functionality in the module in order to increase the efficiency of the system by combining the features, thereby removing excess schedules and allowing fewer and better schedules to proceed in a more timely manner.

18. As per claims 11 and 37, Syswerda teaches wherein said schedule permutation generator includes a third algorithm, said third algorithm:

(a) designating a number of tasks to store in memory at a given time (See column 3, lines 13-30, column 4, lines 20-32, and column 5, lines 51-61, wherein a number of tasks are stored in memory at a given time);

(b) placing the number of tasks in an order of efficiency (See column 4, lines 20-50, and column 6, lines 1-15, wherein the tasks are placed in the order of efficiency); and

(c) determining whether a task outside of the number of tasks is more efficient than a task within the number of tasks (See column 4, lines 20-50 and 60-67, column 5, lines 1-5, and column 6, lines 1-15 and 30-45, wherein the schedules are scored by weighting if a task outside the schedule versus inside the schedule).

19. Claims 12, 13, 38, and 39 are rejected using the same art and rationale as applied above with respect to claims 7 and 8.

20. As per claims 14 and 40, Syswerda teaches wherein said efficiency evaluator includes a fourth algorithm, said fourth enumerating each possible permutation of time dependent tasks (See figures 5 and 7, column 2, lines 40-45, column 3, lines 20-40, column 4, lines 20-36, column 6, lines 1-10, and column 8, lines 1-7, that considers the constraints to determine an ordering of the tasks and tries ever possible permutation of the schedule using the constraints).

21. As per claims 15 and 41, Syswerda teaches wherein said genetic module includes a task list, a genetic schedule permutation generator, and a schedule evaluator (See figure 5, column 3, lines 45-67, column 4, lines 35-67, column 6, lines 32-50, wherein the module has a task list, a permutation generator, and a portion that evaluates the schedule).

22. As per claims 16 and 42, Syswerda teaches wherein said genetic schedule permutation generator includes a fifth algorithm, said fifth algorithm mating a first schedule and a second schedule to breed a third schedule (See column 5, lines 25-35, which discuss crossovers).

23. Claims 17-19 and 43-45 are rejected using the same art and rationale as applied above with regards to claims 7, 8, and 14.

24. As per claims 20 and 46, Syswerda teaches wherein said partitioner module includes a brute force partitioner and a residual evaluator (See figures 5 and 7, column 2, lines 39-45, column 3, lines 20-40 and 50-67, column 4, lines 20-36, column 5, lines 62-67, column 6, lines 1-10, and 14-35, column 8, lines 1-7, wherein tasks are listed in random order using no intelligence or reasoning. A partitioner gives the brute force module the ability to do the scheduling and generate lists of the tasks. See also column 4, lines 20-50 and 60-67, column 5, lines 1-5, and column 6, lines 1-15 and 30-45, wherein the schedules are scored by weighting if a task outside the schedule versus inside the schedule).

25. As per claims 21 and 47, Syswerda discloses a system with a brute force module with a brute force partitioner (See figures 5 and 7, column 2, lines 39-45, column 3, lines 20-40 and 50-67, column 4, lines 20-36, column 5, lines 62-67, column 6, lines 1-10, and 14-35, column 8, lines 1-7, wherein tasks are listed in random order using no intelligence or reasoning. A partitioner gives the brute force module the ability to do the scheduling and generate lists of the tasks). However, Syswerda does not expressly disclose a brute force time completion estimator.

Oba et al. discloses a time completion estimator (See column 2, lines 45-55, and column 4, lines 1-7 and 29-35, wherein the scheduling system has a time completion estimator that estimates the time allowable for scheduling to occur).

Both Syswerda and Oba et al. teach scheduling systems that generate and evaluate schedules to generate optimal schedules based on the constraints. Both Syswerda and Oba et al. also teach running the schedule optimization a number of times at which point the best schedule is chosen. See column 6, lines 32-45, of Syswerda which discusses running the optimization a predetermined number of times and see column 2, lines 45-55, and column 4, lines 1-7 and 29-35, of Oba et al. which discusses running the optimization a predetermined amount of time. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to include a time completion estimator that limited the amount of time for optimization in the system of Syswerda in order to increase the timeliness of the system in generating a schedule for a user by limiting the optimization and evaluation time to a specific interval at which point the best schedule generated thus far is chosen.

26. As per claims 22 and 48, Syswerda teaches wherein said brute force time completion estimator includes a seventh algorithm, said seventh algorithm:

(a) generating a given number of schedule permutations using the brute force module (See figure 5, column 2, lines 39-40, and column 5, lines 62-67, wherein tasks are listed in random order in schedule permutations using no intelligence or reasoning);

However, Syswerda does not expressly disclose steps (b) and (c).

Oba et al. discloses :

(b) estimating a time to complete all possible schedule permutations using the module based on the time taken to generate the given number of schedule permutations (See column 2, lines 45-55, and column 4, lines 1-12 and 29-45, wherein a time is estimated and stored); and

(c) determining whether the estimated time to complete all possible schedule permutations using the module exceeds a given time limit (See column 2, lines 45-55, and column 4, lines 1-12 and 29-45, wherein the estimated time is evaluated).

Both Syswerda and Oba et al. teach scheduling systems that generate and evaluate schedules to generate optimal schedules based on the constraints. Both Syswerda and Oba et al. also teach running the schedule optimization a number of times at which point the best schedule is chosen. See column 6, lines 32-45, of Syswerda which discusses running the optimization a predetermined number of times and see column 2, lines 45-55, and column 4, lines 1-7 and 29-35, of Oba et al. which discusses running the optimization a predetermined amount of time. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to include a time completion estimator that estimates the time needed for schedule optimization and makes determinations concerning said estimated time in the system of Syswerda in order to increase the timeliness of the system in generating a schedule for a user by limiting the optimization and evaluation time to a specific interval at which point the best schedule generated thus far is chosen.

27. As per claims 23 and 49, Syswerda teaches a scheduling system with a deterministic module, a genetic module, and a residual evaluator (See figures 5 and 7, column 2, lines 40-45, column 3, lines 20-40 and 50-67, column 4, lines 20-36 and 50-67, column 6, lines 1-10 and 14-35, and column 8, lines 1-7, that considers the constraints to determine an ordering of the tasks and a genetic algorithm being implemented. See also column 4, lines 20-50 and 60-67, column 5, lines 1-5, and column 6, lines 1-15 and 30-45, wherein the schedules are scored by weighting

if a task outside the schedule versus inside the schedule). However, Syswerda does not expressly disclose a dynamic program module or a time completion estimator in the residual evaluator.

Oba et al. discloses a time completion estimator (See column 2, lines 45-55, and column 4, lines 1-7 and 29-35, wherein the scheduling system has a time completion estimator that estimates the time allowable for scheduling to occur).

Both Syswerda and Oba et al. teach scheduling systems that generate and evaluate schedules to produce optimal schedules based on the constraints. Both Syswerda and Oba et al. also teach running the schedule optimization a number of times at which point the best schedule is chosen. See column 6, lines 32-45, of Syswerda which discusses running the optimization a predetermined number of times and see column 2, lines 45-55, and column 4, lines 1-7 and 29-35, of Oba et al. which discusses running the optimization a predetermined amount of time. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to include a time completion estimator that limited the amount of time for optimization in the system of Syswerda in order to increase the timeliness of the system in generating a schedule for a user by limiting the optimization and evaluation time to a specific interval at which point the best schedule generated thus far is chosen.

Furthermore, Syswerda discloses a deterministically programmed schedule builder that obtains a list of tasks and orders them using hard and soft constraints. Syswerda et al. also discloses when discussing the deterministically programmed schedule builder that this module can include a more complex scheduling techniques, as stated in column 6, lines 5-11, and column 7, lines 29-37. Dynamic programming considers changeable competing or conflicting forces, such as resource and time constraints, to produce multiple possible outcomes and therefore

would be considered a specific type of deterministic scheduling by one of ordinary skill in the art. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to use dynamic programming techniques in the deterministic module of Syswerda in order to increase the ability of the system to build better schedules while still acting deterministically, as stated in column 7, lines 29-40.

28. As per claims 24 and 50, Syswerda teaches a scheduling system including a resource evaluator (See column 4, lines 10-46, which discloses scheduling based on resource constraints and checking to see if the schedule fulfills these constraints). However, Syswerda does not expressly disclose that the resource is hardware.

Oba et al. disclose that the resources being scheduled are hardware (See column 3, lines 55-65, which discloses device resources).

Both Syswerda and Oba et al. teach scheduling systems that generate and evaluate schedules including the use of resource to produce optimal schedules based on the constraints. It would have been obvious to one of ordinary skill in the art at the time of the invention to include hardware as a resource constraining the scheduling of Syswerda in order to increase the efficiency of the system in producing a schedule for a user by including all resources that would be used by said user in the course of the tasks.

29. As per claims 25 and 51, Syswerda discloses a scheduling system wherein said resource evaluator includes a memory module and a central processing unit (See column 4, lines 10-46, and the appendix, which discloses a resource evaluator implemented using a computer system). However, Syswerda does not expressly disclose that the resource is hardware.

Oba et al. disclose that the resources being scheduled are hardware (See column 3, lines 55-65, which discloses device resources).

Both Syswerda and Oba et al. teach scheduling systems that generate and evaluate schedules including the use of resource to produce optimal schedules based on the constraints. It would have been obvious to one of ordinary skill in the art at the time of the invention to include hardware as a resource constraining the scheduling of Syswerda in order to increase the efficiency of the system in producing a schedule for a user by including all resources that would be used by said user in the course of the tasks.

30. As per claims 26 and 52, Syswerda discloses a scheduling system wherein said residual evaluator includes an eighth algorithm, said eighth algorithm:

(a) generating a given number of schedule permutations using the deterministic module (See figures 5 and 7, column 2, lines 40-45, column 3, lines 20-40, column 4, lines 20-36, column 6, lines 1-10, and column 8, lines 1-7, that considers the constraints to determine an ordering of the tasks, which discloses generating a number of schedule combinations).

However, Syswerda does not expressly disclose a dynamic programming module or steps (b) and (c).

Oba et al. discloses :

(b) estimating a time to complete all possible schedule permutations using the module based on the time taken to generate the given number of schedule permutations (See column 2, lines 45-55, and column 4, lines 1-12 and 29-45, wherein a time is estimated and stored); and

(c) determining whether the estimated time to complete all possible schedule permutations using the module exceeds a given time limit (See column 2, lines 45-55, and column 4, lines 1-12 and 29-45, wherein the estimated time is evaluated).

Both Syswerda and Oba et al. teach scheduling systems that generate and evaluate schedules to generate optimal schedules based on the constraints. Both Syswerda and Oba et al. also teach running the schedule optimization a number of times at which point the best schedule is chosen. See column 6, lines 32-45, of Syswerda which discusses running the optimization a predetermined number of times and see column 2, lines 45-55, and column 4, lines 1-7 and 29-35, of Oba et al. which discusses running the optimization a predetermined amount of time. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to include a time completion estimator that estimates the time needed for schedule optimization and makes determinations concerning said estimated time in the system of Syswerda in order to increase the timeliness of the system in generating a schedule for a user by limiting the optimization and evaluation time to a specific interval at which point the best schedule generated thus far is chosen.

Furthermore, Syswerda discloses a deterministically programmed schedule builder that obtains a list of tasks and orders them using hard and soft constraints. Syswerda et al. also discloses when discussing the deterministically programmed schedule builder that this module can include a more complex scheduling techniques, as stated in column 6, lines 5-11, and column 7, lines 29-37. Dynamic programming considers changeable competing or conflicting forces, such as resource and time constraints, to produce multiple possible outcomes and therefore would be considered a specific type of deterministic scheduling by one of ordinary skill in the

art. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to use dynamic programming techniques in the deterministic module of Syswerda in order to increase the ability of the system to build better schedules while still acting deterministically, as stated in column 7, lines 29-40.

31. Claims 53 and 54 are substantially similar to claim 28 and are therefore rejected using the art and rationale set forth above. The constraint file is the input to the partitioner module.

32. Claim 27 is rejected under 35 U.S.C. 103(a) as being unpatentable over Oba et al. (U.S. 5,241,465) in view Syswerda (U.S. 5,319,781).

33. As per claim 27, Oba et al. discloses a dynamic programming module that generates lowest-cost partial paths for a plurality of time dependant tasks, said module adapted to provide at least one solution for a scheduling problem, said dynamic programming module including:

(a) a dynamic program module (See column 2, lines 5-25, wherein the schedule is dynamically generated using a strategy); and

(b) a height-balanced binary tree for providing search insertion and deletion operations on said lowest-cost partial paths (See at least figure 10, column 1, lines 43-45 and 62-67, column 2, lines 1-5, column 4, lines 1-13, column 5, lines 10-34 and 48-65, and column 8, lines 32-65, wherein a height-balanced binary tree provides for search insertion and deletion operations and lowest cost consideration (cost being time, resource, money)).

However, Oba et al. does not expressly disclose hashing function, said hashing function being capable of detecting duplicate solutions generated.

Syswerda discloses the functionality of grouping schedules by score and deleting those schedules grouped at specific score thresholds (See column 4, lines 20-32 and 60-67, column 5, lines 1-5, and column 6, lines 14-40, wherein schedules are evaluated, grouped by score, and deleted at a certain threshold score. Since duplicate schedules would attain the same score when evaluated by the system, duplicates would be grouped together and duplicates would be deleted). However, Syswerda does not specifically disclose a hashing function.

Both Syswerda and Oba et al. teach scheduling systems that generate, evaluate, and eliminate schedules to generate optimal schedules based on the constraints. Specifically, Syswerda teaches grouping schedules by score and deleting those schedules grouped at specific score thresholds. Since hashing functions are well known in the computer programming art and are used to identify and map values to a location, it would have been obvious to one of ordinary skill in the art at the time of the invention to use a hashing function to group the schedules in order to more efficiently map schedules of duplicate task lists and/or duplicate scores to the same location for easy deletion.

Response to Arguments

34. Applicant's arguments with regards to the rejections based on Syswerda (U.S. 5,319,781) and Oba et al. (U.S. 5,241,465) have been fully considered, but they are not persuasive. In the remarks, applicant argues that Syswerda and Oba et al. do not teach or suggest a dynamic programming module that generates lowest-cost partial paths for a plurality of time dependent tasks and that includes a hashing function having a low probability of collision, wherein the hashing function is useable in conjunction with a height balances binary search tree that stores the lowest cost partial paths.

First, Examiner would like to point out that the claims 1 and 28 recite a partitioner module that has the intended use of selecting one of the three modules of enumerative brute force module, genetic module, and dynamic programming module. This partitioner module, if made active, would select one of the three modules to generate a schedule for selecting tasks. The there modules do not specifically recite how they are utilized to accomplish the generation of the schedule. For example, in the broadest reasonable interpretation of the claims, any unit using enumerative brute force to accomplish some scheduling of time dependent tasks would meet the enumerative brute force module claimed. Examiner further points out that the three modules are not active and that if the partitioner module always only selected the genetic module, the dynamic programming module would never occur.

As for claims 53 and 54, Examiner points out that there is no functional connection between the partitioner module and the selecting of the scheduling module. Further, there is no recitation that functionally connects the selected scheduling module with the generation of the schedule. Therefore, if a module is selected, it appears as if no specific activity occurs with this

selected module (i.e. it is merely selected). Furthermore, there is no specific recitation of what the enumerative brute force module and genetic module include and how these modules and the dynamic programming module relate to the rest of the claim. Therefore elements a, b, and c in each of claims 53 and 54 appear to occur independently of each other.

As to Applicant's argument, Examiner respectfully disagrees. As stated above, Syswerda teaches a system that chooses between various available scheduling functions of the system to generate potential schedules. Syswerda specifically teaches that the system selects and uses a genetic module, a enumerative brute force module, or a deterministic function. However, Syswerda does not expressly disclose a dynamic programming module, such as the one claimed by the Applicant. Oba et al. a dynamic programming module that generates lowest-cost partial paths for said plurality of time-dependent tasks with a height-balanced binary tree for providing search insertion and deletion operations. See at least figure 10, column 1, lines 43-45 and 62-67, column 2, lines 1-5, column 4, lines 1-13, column 5, lines 10-34 and 48-65, and column 8, lines 32-65. However, Oba et al. does not expressly disclose a hashing function having a low probability of collisions. Examiner maintains that hashing functions are well known in the computer programming art and are used to identify and map values to a location for faster data retrieval, as discussed above. Therefore, since the claims broadly recite this dynamic programming module with no specific recitation of how it is used to perform scheduling and since this module is not necessarily active, Examiner maintains that the prior art set forth above does teach and suggest the claimed invention.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Beth Van Doren whose telephone number is (703) 305-3882. The examiner can normally be reached on M-F, 8:30-5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tariq Hafiz can be reached on (703) 305-9643. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

bvd
March 10, 2005

TARIQ R. HAFIZ
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 3600